# SEG301 - SEARCH ENGINES & INFORMATION RETRIEVAL

## ĐẶC TẢ ĐỒ ÁN MÔN HỌC (PROJECT SPECIFICATION)

### THÔNG TIN CHUNG

* **Thời lượng:** 10 tuần (Project-Based Learning)
* **Hình thức:** Nhóm 3 sinh viên

### 1. TỔNG QUAN & MỤC TIÊU (OVERVIEW)

Sinh viên sẽ hoá thân thành các Kỹ sư Dữ liệu & AI để xây dựng một **Vertical Search Engine** (Máy tìm kiếm chuyên biệt) từ con số 0.

**Thách thức cốt lõi:**

1. **Big Data:** Xử lý tối thiểu **1.000.000 documents**.
2. **Deep Tech:**
   * *Giai đoạn 1 (Hardcore):* Tự lập trình Crawler, Indexer (SPIMI), Ranker (BM25).
   * *Giai đoạn 2 (Modern):* Tích hợp AI (Vector Search, LLM).

### 2. QUY ĐỊNH BẮT BUỘC (ZERO TOLERANCE POLICY)

Vi phạm các điều sau sẽ nhận **0 điểm** toàn bộ Project:

1. **GITHUB HISTORY:** Toàn bộ quá trình làm việc phải có lịch sử commit rõ ràng từ tuần đầu tiên. Không chấp nhận việc "Code một lần rồi upload hết".
2. **AI LOGGING:** Phải nộp kèm file ai\_log.md chứa lịch sử chat với AI. Nếu code trong bài hoàn hảo nhưng không có log tương ứng -> Coi như gian lận.

### 3. CẤU TRÚC ĐIỂM SỐ (GRADING SCHEME)

* **60% - Project:** Chia đều cho 3 Milestones.
* **40% - Final Exam:** Thi lý thuyết tập trung.

### 4. CHI TIẾT CÁC CỘT MỐC & TIÊU CHÍ ĐÁNH GIÁ

#### MILESTONE 1: DATA ACQUISITION (20%) - Deadline: Tuần 4

*Mục tiêu: Xây dựng bộ dữ liệu sạch 1 triệu dòng.*

**A. Các công việc cụ thể cần làm:**

1. **Crawl:** Viết script Python (dùng requests, aiohttp, selenium) để thu thập dữ liệu. Bắt buộc xử lý đa luồng (Multi-threading) hoặc Bất đồng bộ (Async) để đảm bảo tốc độ.
2. **Clean:**
   * Loại bỏ thẻ HTML, script rác.
   * Tách từ tiếng Việt (Word Segmentation) dùng thư viện (PyVi, Underthesea).
   * Xử lý các trường hợp trùng lặp (De-duplication).
3. **Storage:** Lưu trữ dữ liệu dưới dạng cấu trúc (JSONL hoặc Parquet). Không lưu 1 triệu file txt rời rạc.

**B. Tiêu chí đánh giá (Thang 10 -> quy đổi 20%):**

* **(4đ) Khối lượng & Chất lượng dữ liệu:** Đủ 1.000.000 docs. Dữ liệu sạch, không lỗi font, đã tách từ.
* **(3đ) Kỹ thuật Crawl:** Code chạy được, có xử lý Async/Multi-thread, có cơ chế Resume (chạy tiếp khi rớt mạng).
* **(2đ) GitHub & Log:** Commit đều đặn, log AI đầy đủ.
* **(1đ) Insight:** Có báo cáo thống kê về dữ liệu (số lượng từ vựng, độ dài trung bình docs).

#### MILESTONE 2: CORE SEARCH ENGINE (20%) - Deadline: Tuần 7

*Mục tiêu: Hiểu bản chất thuật toán Indexing & Ranking.*

**A. Các công việc cụ thể cần làm:**

1. **Indexing (Hardcore):** Code tay thuật toán **SPIMI** (Single-Pass In-Memory Indexing).
   * Chia 1 triệu docs thành các block nhỏ.
   * Index từng block trên RAM -> Ghi xuống đĩa.
   * Merge các block lại thành file Inverted Index hoàn chỉnh.
2. **Ranking (Hardcore):** Code tay thuật toán **BM25**.
   * Tự tính TF, IDF, Average Document Length.
   * Không được gọi hàm rank() của thư viện có sẵn.
3. **Console App:** Viết chương trình chạy dòng lệnh cho phép nhập từ khóa và trả về kết quả top 10.

**B. Tiêu chí đánh giá (Thang 10 -> quy đổi 20%):**

* **(4đ) Thuật toán SPIMI:** Implement đúng logic SPIMI. Chạy index 1 triệu docs không bị tràn RAM (Memory Error).
* **(3đ) Thuật toán BM25:** Kết quả trả về hợp lý (Document chứa từ khóa nhiều và hiếm phải lên top).
* **(2đ) Hiệu năng:** Tốc độ trả về kết quả tìm kiếm **< 1 giây**.
* **(1đ) Demo:** Trả lời tốt các câu hỏi vấn đáp về code ("Tại sao dòng này lại viết thế này?").

#### MILESTONE 3: FINAL PRODUCT (20%) - Deadline: Tuần 10

*Mục tiêu: Sản phẩm thực tế & Ứng dụng AI.*

**A. Các công việc cụ thể cần làm:**

1. **Vector Search:** Sử dụng thư viện (FAISS/ChromaDB) và Model Embedding (Sentence-Transformers/PhoBERT) để index lại dữ liệu theo ngữ nghĩa.
2. **Web Interface:** Xây dựng giao diện web (Streamlit/Flask/React) thân thiện.
3. **Hybrid Search:** Kết hợp kết quả từ BM25 (M2) và Vector Search (M3) để ra kết quả tối ưu.
4. **Evaluation:** Chạy bộ test (khoảng 20 queries) để tính chỉ số Precision@10, so sánh giữa Search thường và AI Search.

**B. Tiêu chí đánh giá (Thang 10 -> quy đổi 20%):**

* **(3đ) Tính năng AI:** Tích hợp thành công Vector Search. Tìm được các query ngữ nghĩa (Ví dụ: Search "máy tính chơi game" ra kết quả chứa "laptop gaming" dù không khớp chữ).
* **(3đ) Sản phẩm Web:** Giao diện đẹp, đầy đủ tính năng (Search, Filter, Pagination), không lỗi crash.
* **(2đ) Đánh giá (Evaluation):** Có bảng so sánh Precision/Recall và phân tích tại sao AI tốt hơn/tệ hơn trong từng trường hợp.
* **(2đ) Kỹ năng trình bày:** Slide rõ ràng, demo suôn sẻ, log AI minh bạch.

### 5. QUY ĐỊNH VỀ NHẬT KÝ AI (ai\_log.md)

Để giảm tải áp lực format, sinh viên sử dụng file Markdown đơn giản.

**Yêu cầu:**

* Copy toàn bộ lịch sử chat (Prompt của bạn và Câu trả lời của AI) vào file ai\_log.md.
* Phân chia theo ngày tháng.
* **Mục đích:** Giảng viên sẽ sử dụng công cụ AI để đọc file này, tóm tắt lại quá trình làm việc của bạn và đối chiếu với code trên GitHub để xác minh tính trung thực (Author Verification).

**Mẫu file ai\_log.md:**

# AI INTERACTION LOG  
  
## Date: 2025-10-15  
\*\*Task:\*\* Debug lỗi SPIMI bị tràn RAM  
  
\*\*User:\*\* Tôi đang code thuật toán SPIMI bằng Python. Khi dictionary đạt 500MB thì tôi write xuống đĩa, nhưng tại sao RAM vẫn không giảm? Đây là code của tôi: [Paste code]  
  
\*\*AI (ChatGPT):\*\* Vấn đề nằm ở bộ thu gom rác (Garbage Collector) của Python. Dù bạn `del dictionary` nhưng Python chưa giải phóng ngay. Bạn nên dùng `gc.collect()` sau khi xóa...  
  
## Date: 2025-10-20  
\*\*Task:\*\* Viết regex cho Crawler  
...

### 6. CHỦ ĐỀ DỰ ÁN (CHỌN 1 TRONG 4)

*Yêu cầu chung: 1.000.000 Documents*

1. **Bất động sản:** Xử lý tin trùng lặp. (Nguồn: Batdongsan, Chotot).
2. **Thương mại điện tử:** So sánh giá đa sàn. (Nguồn: Shopee, Tiki).
3. **Thông tin Doanh nghiệp:** Merge dữ liệu Thuế & Review. (Nguồn: Masothue, Reviewcongty).
4. **Social Listening:** Xử lý ngôn ngữ mạng/Teencode. (Nguồn: Voz, TinhTe).

### **7. CẤU TRÚC THƯ MỤC GITHUB (GITHUB REPOSITORY STRUCTURE)**

Yêu cầu bắt buộc về cách tổ chức file trên GitHub để giảng viên dễ dàng kiểm tra và chấm điểm.

SEG301-Project-GroupX/  
├── .gitignore # File cấu hình git ignore (bỏ qua venv, data lớn, \_\_pycache\_\_)  
├── README.md # Hướng dẫn cài đặt, chạy dự án & Link tải Full Dataset  
├── ai\_log.md # Nhật ký sử dụng AI (Cập nhật liên tục theo ngày)  
├── requirements.txt # Danh sách các thư viện cần thiết (pip freeze > requirements.txt)  
├── docs/ # Thư mục chứa báo cáo (PDF/Markdown)  
│ ├── Milestone1\_Report.pdf  
│ ├── Milestone2\_Report.pdf  
│ └── Milestone3\_Presentation.pdf  
├── data\_sample/ # Chỉ chứa khoảng 100-500 docs mẫu để test (KHÔNG UP 1 TRIỆU DOCS)  
│ └── sample.jsonl  
├── src/ # Source code chính  
│ ├── \_\_init\_\_.py  
│ ├── crawler/ # Milestone 1: Code thu thập dữ liệu  
│ │ ├── spider.py # Logic crawl chính  
│ │ ├── parser.py # Xử lý HTML, tách từ  
│ │ └── utils.py # Hàm phụ trợ (Proxy, User-agent)  
│ ├── indexer/ # Milestone 2: Code tạo chỉ mục  
│ │ ├── spimi.py # Thuật toán SPIMI  
│ │ ├── merging.py # Logic merge block  
│ │ └── compression.py # Nén dữ liệu (nếu có)  
│ ├── ranking/ # Milestone 2 & 3: Code xếp hạng  
│ │ ├── bm25.py # Thuật toán BM25 (Code tay)  
│ │ └── vector.py # Semantic Search (Dùng thư viện cho M3)  
│ └── ui/ # Milestone 3: Giao diện người dùng  
│ └── app.py # Streamlit/Flask app  
└── tests/ # (Khuyến khích) Unit tests cho các thuật toán core  
 ├── test\_spimi.py  
 └── test\_bm25.py

**Lưu ý quan trọng:**

1. **.gitignore**: Bắt buộc phải có để tránh upload rác (file .DS\_Store, \_\_pycache\_\_) hoặc dữ liệu quá nặng.
2. **data\_sample/**: Tuyệt đối **không upload file dữ liệu 1 triệu dòng** lên GitHub (sẽ bị lỗi limit). Hãy upload file full lên Drive/OneDrive và để link trong README.md.
3. **src/**: Code phải được module hóa (chia thành các file nhỏ) thay vì dồn hết vào 1 file main.py.

**CHÚC CÁC BẠN HOÀN THÀNH TỐT NHIỆM VỤ!**